Docket No.: 103488-0003
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Colin P. Britton et al.

Application No.: 09/917,264 Confirmation No.: 7813

Filed: July 27, 2001 Art Unit: 2175

For: METHODS AND APPARATUS FOR ENTERPRISE APPLICATION INTEGRATION Examiner: J. N. Abel

APPEAL BRIEF

MS Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

The Applicants appeal to the Board of Patent Appeals and Interferences (the "Board") from the Examiner's rejections of claims 27-30, 34, 35, 39-41, 43, 44, and 53 of the above-cited application. A notice to this effect was filed on August 19, 2004.

Pursuant to 37 C.F.R § 41.37(a), a fees in the amount of $170.00 is filed herewith. Please charge any additional fees to Deposit Order Account 14-1449. A copy of the pending claims is filed as in the Appendix hereto.

REAL PARTY OF INTEREST

The real party in interest in this appeal is Metatomix, Inc.

RELATED APPEALS AND INTERFERENCES

There are no other appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

STATUS OF CLAIMS

Claims 27-30, 34, 35, 39-41, 43, 44, and 53 stand rejected. It is the rejection of these claims that the Applicants appeal. A listing of the claims is provided in the Appendix hereto.

STATUS OF AMENDMENTS

Claims 36-38 were canceled and claims 34 and 41 were amended in a response filed on August 19, 2004, subsequent to the Final Office Action mailed on April 19, 2004. These amendments were entered by the Examiner as reflected in the Advisory Action dated September 30, 2004.

SUMMARY OF CLAIMED SUBJECT MATTER

The claimed invention is directed to an enterprise application integration system.

Per pending claim 27, the claimed invention provides for digital data processing methods for enterprise application integration which including storing, in a data store (for example, see specification at p. 7, line 9 et seq.), RDF triples representing transactional information received from each of a plurality of databases (e.g., specification at p. 3, lines 14-17). The claim further recites displaying, on a browser (e.g., Fig. 1, element 118, and specification at p. 6, line 9), a markup language document (e.g., specification at p.8, line 25 et seq.) – which browser markup language document generates one or more queries (id.) for application to the data store in response to one or more user selections and/or responses to user-input controls specified by the document and which presents via the browser content generated from the data store in response to those queries (id.).

Pending claim 39 parallels claim 27, additionally reciting generating a directed graph from the RDF triplets (e.g., specification at p. 11, line 11 et seq.).

Pending claim 43 also parallels claim 27, additionally reciting storing in the data store one or more further queries for application to at least one of the databases (e.g., specification at p. 12, line 1 et seq.), applying any of the queries to one or more of the plurality of databases using respective applications program interfaces, retrieving information from one or more databases in response to the applied query (id.), converting that retrieved information into said RDF triplets (id.), responding to an applied query by generating a directed graph from the RDF triplets (id.), and parsing the directed graph and presenting content generated therefrom via the browser (e.g., specification at p. 11, line 11 et seq.).

Per pending claim 53, the claimed invention provides for a digital data processing method for enterprise application integration including removing redundancies in RDF triples (e.g., specification at p. 10, line 26 et seq.) by executing the steps of (i) comparing sequential levels of objects of RDF triples, (ii) determining a confidence level that two or more triplets at a compared level represent redundant information, and (iii) merging into a bad triplets determined to be redundant on a basis of that confidence level.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The issues presented for review in this appeal are:

(1) Whether independent claim 53 is anticipated by Lipkin et al., U.S. Patent Application Pub. No. 2002/0049788 A1 ("Lipkin");

(2) Whether independent claims 27, 39, and 43, and dependent claims 28-30, 34, 40, 41, and 44 are unpatentable over Lipkin in view of Bradford, U.S. Patent No. 6,678,679 ("Bradford"); and

(3) Whether independent claim 43, and dependent claims 35 and 44 are unpatentable over Lipkin in view of Bradford in further view of Delcambre et al, U.S. Pub. No. 2002/0059566 A1 ("Delcambre").

ARGUMENTS

I. Independent Claim 53 Is Not Anticipated By Lipkin

Claim 53 is directed to a digital data processing method for enterprise application integration including removing redundancies in RDF triples by executing the steps of (i) comparing sequential levels of objects of RDF triples, (ii) determining a confidence level that two or more triplets at a compared level represent redundant information, and (iii) merging into a bad triplets determined to be redundant on a basis of that confidence level.

Lipkin fails to teach these steps. Specifically, with respect to that part of claim 53 which recites "determining a confidence level that two or more triplets at a compared level represent redundant information", the Examiner cites Lipkin at p. 71, col. 1, lines 27-col. 2, line 25. That text is reprinted below. Nowhere does it suggest removing redundancy – much less, doing so

based on confidence levels. Indeed, quite to the contrary, in the cited passage, Lipkin laments that his chosen schema is redundant, yet he proposes no solution:

[1103] Database schema

[1104] The database schema used has two main advantages:

[1105] 1. Simplicity. All RDF data is stored in a single table and all SQL is written to read and write to this table.

[1106] 2. Support for non-RDF data. It is simple to cast non-RDF data into this format so that existing or legacy data can be queried by the DatabaseMR using RQL.

[1107] So, for example, for the following example data stored in an "invoices" table:

id	last_updated	customer
1	10-JAN-99	Ford
2	25-FEB-99	Cisco

[1108] The view used by the MR can be augmented as followed to incorporate this data:

```
create view invoice_date_triples as
select 'last_updated' "rdf_property",
       ('invoice#' || id) "rdf_resource",
       to_char(last_updated, 'YYYY-MM-DD') "rdf_object"
       from test_invoices;
create view invoice_customer_triples as
select 'customer' "rdf_property",
       ('invoice#' || id) "rdf_resource",
       customer "rdf_object"
       from test_invoices;
drop view MR_triples;
create view MR_triples as
   (select rdf_property, rdf_resource, rdf_object from
invoice_date_triples)
    union
   (select rdf_property, rdf_resource, rdf_object from
invoice_customer_triples)
    union
   (select rdf_property, rdf_resource, rdf_object from
MR_triples_base);
```

[1109] This will result in the following additional triples being available from the MR:

rdf_resource	rdf_property	rdf_object
invoice#1	last_updated	10-JAN-99
invoice#1	customer	Ford
invoice#2	last_updated	25-FEB-99
invoice#2	customer	Cisco

[1110] The disadvantage to this schema is that it is not normalized and stores a tremendous amount of duplicate data. Many values for rdf resource and rdf property will be duplicated, since the same resource will have a number of properties, and property names will come from a well-known set.

Lipkin, p.71, col.1, lines 27-67

[1111] RQLParser

[1112] RQLParser parses an RQL document and builds an execution plan for the query. The plan consists of a tree of Java classes called "Operators," where each Operator is responsible for returning a Vector of matching resources.

[1113] The Operator interface is defined as follows:

```
public interface Operator
{
/**
 * An operator knows how to return a Vector of matching
resource values
 * (typically URIs).
 * @param conn JDBC connection to the MR
 * @param targetRDF Target RDF file.
 * @return Vector of matching resources
 * @exception SQLException Thrown on a database error
 */
  public Vector getResources(Connection conn, String
targetRDF) throws SQLException, ParseException;
} /* Operator */
```

Lipkin, p. 71, col. 2, lines 1-25

Specifically, referring to the paragraph bridging column 1 and 2 on page 71, above, Lipkin states, "[t]he disadvantage to this schema is that it is not normalized and stores a tremendous amount of duplicate data. Many values for rdf resource and rdf property will be

duplicated, since the same resource will have a number of properties, and property names will come from a well known set." This passage thus does not, contrary to the Examiner's assertion, teach or suggest removing redundancy based on confidence levels.

Nor, with respect to that part of claim 53 which recites "merging into a bag triplets determined to be redundant on a basis of that confidence level" do the passages from Lipkin cited by the Examiner provide relevant teachings. The cited passages, page 19, col. 2, lines 14-22, and page 66, col. 1, lines 10-21, are reprinted below. These do not teach removing redundancy. Nor do they teach merging redundant triplets into a bag (or container or any other "collection"):

[0385] The bean is automatically saved to the persistent store when it is created by a client application using the create() method, and when the container decides to synchronize the bean's state with the database if the bean's data has been changed by the client application. The container's decision is based on such factors as transactions, concurrency, and resource management. The container will remove the data from persistent store when the remove() method is called by a client on an entity bean.

Lipkin, p. 19, col. 2, lines 14-22

[1023] Property

[1024] The Property element identifies a specific, named property of a Resource. Its contents identify the named property (also known as the predicate). Its contents can be a nested property, that is, multiple property names separated by forward slashes. This syntax may navigate over multiple properties, where each property value is a resource with its own properties. This may be the same syntax used by RDF Query's "path" attribute for nested queries.

[1025] As a convenience, it may not be necessary to specify Container-related properties as part of the path, that is, Bag, Seq, Alt, and li elements are automatically navigated past.

Lipkin, p. 66, col. 1, lines 10-21

As evident upon review of these passages, there is simply nothing in them that pertains to redundancy reduction – much less, removing redundancy based, in part, on merging triplets determined to be redundant into a bag or other collection.

Likewise, with respect to that part of claim 53 which recites "comparing sequential levels of objects of the RDF triples," the passage from Lipkin cited by the Examiner provides no relevant teachings. The cited passage, page 9, col. 2, lines 43-67, is reprinted below. This does not teach removing redundancy based on comparing sequential levels of objects of RDF triples, or otherwise. Indeed, although the passage uses the word "object" and displays a table with levels, nothing it states has to do with removing redundancy, at levels or otherwise:

[0241] As an example, the following are the values for a class of business object representing domains:

id	ui_name	description	enumber	insert_spid
ddcls000000000001095	Domain	Hierarchal Domain	195	10560
update_spid	delete_spid	sel_det_spid	finder_id	fixed_attr_ct
10562	10561	10563	15710	14
attr_ct	flags	next_attr_enum	prefix	table_name
14	1100001100	100000	domin	fgt_domain
domain_enum	java_class_name		hlevel	parent_id
	com.saba.busobj.SabaDomain		1	

[0242] 2. fgt_dd_attr

[0243] The attributes of each class of business object is stored in this table. This table also describes basic properties of each attribute.

Lipkin, p. 9, col. 2, lines 47-67

For the foregoing reasons, among others, Lipkin does not anticipate independent claim 53.

II. Independent Claims 27 and Dependent Claims 28-30 and 34 Are Patentable Over Lipkin In View of Bradford

Independent claim 27 is directed to digital data processing methods for enterprise application integration which includes storing, in a data store, RDF triplets representing transactional information received from each of a plurality of databases. The claim recites displaying on a browser a markup language document – which *browser markup language document* generates one or more queries for application to the data store in response to one or more user selections and/or responses to user-input controls specified by the document. The markup language document also presents, via the browser, content generated from the data store in response to those queries.

The cited references fail to teach such use of a markup language document.

Lipkin is directed to a web content platform with a repository of metadata in RDF form imported from external sources (see p. 62, paragraph 938). Nowhere does Lipkin teach or suggest displaying on a browser a markup language document that generates one or more queries for application to the data store in response to one or more user selections and/or responses to

user-input controls specified by the document and that presents via the browser content generated from the data store in response to those queries. Indeed, although Lipkin does suggest the use of a web browser in connection with his disclosed system, nowhere does that application suggest use of that browser with a markup language document having these and the other features recited in the pending claims.

Notwithstanding that above, the Examiner has attempted to re-read claim 27 onto Lipkin, urging that Applicant's recitation "markup language documents" reads on Lipkin's XML. *See*, Office Action at page 6, first paragraph. That misses the point.

The Applicants do not deny that Lipkin uses XML. They deny that Lipkin's XML equates with the markup language document recited in claim 27, specifically, a markup language document that:

(i) is displayed on a browser

(ii) generates one or more queries for application to the data store,

(iii) in response to one or more user selections and/or responses to user-input controls specified *by the document*, and,

(iv) presents, via the browser, content generated from the data store in response to those queries.

Lipkin's XML does little if any of this. And, although the Examiner attempts to cite passages to the contrary, review of those passages reveal that they have little to do with this topic. For example, the cited passage at page 73, col. 2, lines 1-25, does not teach used of a markup language to generate queries. It merely discusses the use of XML to transfer messages from a so-called Business Server to so-called Interface Server for display in HTML or on Palm Pilots (or other PDAs). Nowhere does Lipkin mention *using the XML* to generate queries:

> **[0024]** In another embodiment, the present invention provides a network node in a network having a user node including a browser program or other general-purpose or purpose-built client program, coupled to the network, the user node providing information and requests for information, and providing application related commands on the network, the network node including a server node responsive to a request from the user node to process data for generating web content, whereby the server node provides a first mechanism for generating a plurality of tasks required to separate data production elements, interactive elements, and display elements, and a second mechanism for generating the web content based on the separated elements.

Put another way, unlike Applicant's recited markup language document, Lipkin's XML is merely the messenger – not the means. Applicant's recited markup language document, on the other hand, is the *means*, i.e., the means for generating queries.

Bradford does not provide teachings that remedy the shortcomings of Lipkin. Thus, Bradford does not teach or suggest a browser markup language document that generates one or more queries for application to the data store in response to one or more user selections and/or responses to user-input controls specified by the document and that presents via the browser content generated from the data store in response to those queries. The Examiner cites Bradford only for the proposition that transactional information can be stored in RDF. However, this teaching does not rise to the level of a browser markup language document of the type recited by Applicant in independent claim 27.

For the foregoing reasons, among others, Lipkin, in view of Bradford, does not anticipate independent claim 27, and dependent claims 28-30 and 34.

III. Independent Claims 39 and Dependent Claims 40 and 41 Are Patentable Over Lipkin In View of Bradford

Independent claim 39 is directed to digital data processing methods for enterprise application integration which including storing, in a data store, RDF triples representing transactional information received from each of a plurality of databases, displaying on a browser a markup language document – which *browser markup language document* generates one or more queries for application to the data store in response to one or more user selections and/or responses to user-input controls specified by the document. The markup language document presents, via the browser, content generated from the data store in response to those queries. The method further includes generating a directed graph from the RDF triplets.

The cited references fail to teach such a use of a markup language document.

The Applicants do not deny that Lipkin uses XML. They deny that Lipkin's XML equates with the markup language document recited in claim 39, specifically, a markup language document that:

(i) is displayed on a browser

(ii) generates one or more queries for application to the data store,

(iii) in response to one or more user selections and/or responses to user-input controls specified *by the document*, and,

(iv) presents, via the browser, content generated from the data store in response to those queries, and

(v) generates a directed graph from the RDF triplets

Lipkin's XML does little if any of this. And, although the Examiner cites passage to the contrary, review of those passages reveal that they have little to do with this topic. For example, the cited passage at page 73, col. 2, lines 1-25, does not teach used of a markup language to generate queries. It merely discusses the use of XML to transfer messages from a so-called Business Server to so-called Interface Server for display in HTML or on Palm Pilots (or other PDAs). Nowhere does Lipkin mention *using the XML* to generate queries (see passage from Lipkin above).

Put another way, unlike Applicant's recited markup language document, Lipkin's XML is merely the messenger – not the means. Applicant's recited markup language document, on the other hand, is the *means*, i.e., the means for generating queries.

Bradford does not provide teachings that remedy the shortcomings of Lipkin. Thus, Bradford does not teach or suggest a browser markup language document that generates one or more queries for application to the data store in response to one or more user selections and/or responses to user-input controls specified by the document and that presents via the browser content generated from the data store in response to those queries. The Examiner cites Bradford only for the proposition that transactional information can be stored in RDF. However, this

teaching does not rise to the level of a browser markup language document of the type recited by Applicant in independent claim 39.

For the foregoing reasons, among others, Lipkin, in view of Bradford, does not anticipate independent claim 39, and dependent claim 40 and 41.

IV. Independent Claims 43 and Dependent Claim 44 Are Patentable Over Lipkin In View of Bradford

Claim 43 is directed to digital data processing methods for enterprise application integration which including storing, in a data store, RDF triples representing transactional information received from each of a plurality of databases, and displaying on a browser a markup language document – which *browser markup language document* generates one or more queries for application to the data store in response to one or more user selections and/or responses to user-input controls specified by the document and which presents via the browser content generated from the data store in response to those queries. Further, the method includes storing in the data store one or more further queries for application to at least one of the databases, applying any of the queries to one or more of the plurality of databases using respective applications program interfaces, retrieving information from one or more databases in response to the applied query, converting that retrieved information into said RDF triplets, responding to an applied query by generating a directed graph from the RDF triplets, and parsing the directed graph and presenting content generated therefrom via the browser.

The cited references fail to teach such a use of a markup language document.

The Applicants do not deny that Lipkin uses XML. They deny that Lipkin's XML equates with the markup language document recited in claim 43, specifically, a markup language document that:

(i) is displayed on a browser

(ii) generates one or more queries for application to the data store,

(iii) in response to one or more user selections and/or responses to user-input controls specified *by the document*, and,

(iv) presents, via the browser, content generated from the data store in response to those queries,

(v) stores in the data store one or more further queries for application to at least one of the databases,

(vi) applies any of said queries to one or more of the plurality of databases using respective applications program interfaces,

(vii) retrieves information from one or more databases in response to the applied query,

(viii) converts that retrieved information into said RDF triplets,

(ix) responds to an applied query by generating a directed graph from the RDF triplets, and

(x) parses the directed graph and presents content generated therefrom via the browser.

Lipkin's XML does little if any of this. And, although the Examiner attempts to cite passages to the contrary, review of those passages reveal that they have little to do with this topic. For example, the cited passage at page 73, col. 2, lines 1-25, does not teach used of a markup language to generate queries. It merely discusses the use of XML to transfer messages from a so-called Business Server to so-called Interface Server for display in HTML or on Palm Pilots (or other PDAs). Nowhere does Lipkin mention *using the XML* to generate queries (see passage from Lipkin above).

Put another way, unlike Applicant's recited markup language document, Lipkin's XML is merely the messenger – not the means. Applicant's recited markup language document, on the other hand, is the *means*, i.e., the means for generating queries.

Bradford does not provide teachings that remedy the shortcomings of Lipkin. Thus, Bradford does not teach or suggest a browser markup language document that generates one or more queries for application to the data store in response to one or more user selections and/or responses to user-input controls specified by the document and that presents via the browser

content generated from the data store in response to those queries. The Examiner cites Bradford only for the proposition that transactional information can be stored in RDF. However, this teaching does not rise to the level of a browser markup language document of the type recited by Applicant in independent claim 43.

For the foregoing reasons, among others, Lipkin, in view of Bradford, does not anticipate independent claim 43 and dependent claim 44.

V. Independent Claim 43 and Dependent Claims 35 and 44 are Patentable Over Lipkin In View of Bradford In Further View of Delcambre

Claim 35 depends on independent claim 27, and claim 44 depends on independent claim 43.

As discussed above, independent claims 27 and 44 are patenable over Lipkin for various reasons. Neither Bradford not Delcambre provide teachings that remedy the shortcomings of Lipkin. Thus, neither of those secondary references teaches or suggests a browser markup language document that generated one or more queries for application to the data store in response to one of more user selections and/or responses to user-input controls specified by the document and that presents, via the browser, content generated from the data store in response to those queries.

Indeed, the Examiner cites Bradford only for the proposition that transactional information can be stored in RDF. The Examiner cites Delcambre for the proposition that a data query can be stored. However, none of those teachings rises to the level of browser markup language document of the type recited by the Applicants in independent claim 43.

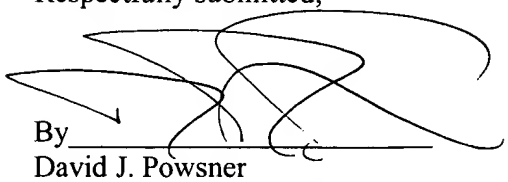
For the foregoing reasons, among others, Lipkin, in view of Bradford, in further view of Delcambre, does not anticipate independent claim 43 and dependent claim 35 and 44.

CONCLUSION

For the reasons detailed above, the Applicants-Appellants respectfully request that the Board reverse the Examiner's rejections.

Dated: October 18, 2004 11-18-04

Respectfully submitted,



By_____
David J. Powsner
Registration No.: 31,868
NUTTER MCCLENNEN & FISH LLP
World Trade Center West
155 Seaport Boulevard
Boston, Massachusetts 02210-2604
(617) 439-2000
(617) 310-9000 (Fax)
Attorney for Applicant

APPENDIX A

Claims Involved in the Appeal of Application Serial No. 09/917,264

Claims 1-26 (cancelled).

27. (previously amended) A digital data processing method for enterprise application integration comprising

storing, in a data store, RDF triplets representing transactional information received from each of a plurality of databases,

displaying on a browser a markup language document that

(i) generates one or more queries for application to the data store,

(ii) presents, via the browser, content generated from the data store in response to the one or more queries,

where the markup language document identifies user interface components to display said content.

28. (previously amended) A method according to claim 27, wherein at least one of the databases stores additional data in a form other than as RDF triplets.

29. (previously amended) A method according to claim 27, wherein the markup language document identifies the queries to be generated in response to one or more user selections and/or responses to user-input controls specified by that document.

30. (previously amended) A method according to claim 27, wherein the markup language document identifies one or more menus, button bars or other controls that allow the user to specify a search or otherwise modify the content presented via the browser.

Claims 31 - 33 (cancelled).

34. (previously amended) A method according to claim 28, wherein the storing step includes storing an RDF triplet representing any of marketing information or an e-commerce or other transaction.

35. (previously amended) A method according to claim 27, comprising

storing in the data store a query for application to at least one of the databases

applying the query to one or more of the plurality of databases using respective applications program interfaces ("API"),

retrieving information from the one or more databases in response to the applied query,

converting that retrieved information into said RDF triplets.

Claims 36 – 38 (cancelled)

39. (previously amended) A digital data processing method for enterprise application integration comprising

storing, in a data store, RDF triplets representing transactional information received from each of a plurality of databases,

displaying on a browser a markup language document that

(i) generates one or more queries for application to the data store in response to one or more user selections and/or responses to user-input controls specified by that document,

(ii) presents, via the browser, content generated from the data store in response to the one or more queries,

where the markup language document identifies user interface components to display said content.

generating a directed graph from the RDF triplets.

40. (previously amended) A method according to claim 39, comprising the step of generating the directed graph in response to a said query.

41. (previously amended) A method according to claim 40, wherein the directed graph reflects any of marketing information or an e-commerce or other transaction.

Claim 42 (cancelled).

43. (previously amended) A digital data processing method for enterprise application integration comprising

storing, in a data store, RDF triplets representing transactional information received from each of a plurality of databases,

displaying on a browser a markup language document that

(i) generates one or more queries for application to the data store in response to one or more user selections and/or responses to user-input controls specified by that document,

(ii) presents, via the browser, content generated from the data store in response to the one or more queries,

where the markup language document identifies user interface components to display said content,

storing in the data store one or more further queries for application to at least one of the databases,

applying any of said queries to one or more of the plurality of databases using respective applications program interfaces ("API"),

retrieving information from the one or more databases in response to the applied query,

converting that retrieved information into said RDF triplets.

responding to an applied query by generating a directed graph from the RDF triplets,

parsing the directed graph and presenting content generated therefrom via the browser .

44. (Original) A method according to claim 43 wherein the parsing step includes parsing the directed graph and presenting therefrom consolidated information plural ones of the databases.

Claims 45 - 52 (cancelled).

53 (Previously Added) A digital data processing method for enterprise application integration comprising

removing redundancies in the RDF triples by executing the steps of

i) comparing sequential levels of objects of the RDF triples,

ii) determining a confidence level that two or more triplets at a compared level represent redundant information,

iii) merging into a bag triplets determined to be redundant on a basis of that confidence level.